Journal of Korea Multimedia Society Vol. 20, No. 4, April 2017(pp. 649-659)
https://doi.org/10.9717/kmms.2017.20.4.649

방향성이 있는 동적인 도로에서 실시간 최단 경로 탐색 시스템의 설계와 구현

권오성†, 조형주††

Design and Implementation of Real-time Shortest Path Search System in Directed and Dynamic Roads

Oh-Seong Kwon†, Hyung-Ju Cho††

ABSTRACT

Typically, a smart car is equipped with access to the Internet and a wireless local area network. Moreover, a smart car is equipped with a global positioning system (GPS) based navigation system that presents a map to a user for recommending the shortest path to a desired destination. This paper presents the design and implementation of a real-time shortest path search system for directed and dynamic roads. Herein, we attempt to simulate real-world road environments, while considering changes in the ratio of directed roads and in road conditions, such as traffic accidents and congestions. Further, we analyze the effect of the ratio of directed roads and road conditions on the communication cost between the server and vehicles and the arrival times of vehicles. In this study, we compare and analyze distance-based shortest path algorithms and driving time-based shortest path algorithms while varying the number of vehicles to search for the shortest path, road conditions, and ratio of directed roads.

Key words: Real-time Shortest Path, Directed and Dynamic Roads, Distance-based Shortest Path, Time-based Shortest Path

1. 서 론

최근 자동차 산업이 정보통신기술과 결합하여 주목 받고 있다. 대표적인 제조업인 자동차 산업은 정보통신기술과 융합하면서 빠르게 스마트화가 진행되고 있는 분야이다. 이를 커넥티드 카(connected car)라고도 부르며 흔히 스마트 자동차라 부른다. 스마트 자동차는 네트워크에 접속이 가능하고 무선통신을 통해 차량 내부와 외부 네트워크가 상호 연결되는 물리적 시스템을 갖춘 자동차를 말한다. 스마트 자동차에는 GPS (Global Positioning System)[1] 수신기를 통해 정확한 위도와 경도 정보를 수신하는 내비게이션 시스템이 장착되어있다. 내비게이션 시스템은 사용자가 목적지를 입력하면 모니터에 정밀한 지도를 제시하고, 현재 위치에서 목적지까지 최단경로를 안내하는 기능이 들어있다. 이 때, 사용하는 최단경로를 계산하는 알고리즘으로 Dijkstra[2] 알고리즘과 A-Star(A*)[3] 알고리즘을 주로 사용한다[4,5]. Dijkstra 알고리즘은 가장 짧은 경로를 항상 찾을 수 있다는 장점이 있지만, 모든 경로를 탐색해야

※ Corresponding Author : Hyung-Ju Cho, Address: (37224) Gyeongsang-daero 2559, Sangju-si, Gyeongsang-buk-do, Republic of Korea, TEL : +82-54-530-1455, FAX : +82-54-530-1459, E-mail : hyungju@knu.ac.kr
Receipt date : Nov. 6, 2016, Revision date : Jan. 22, 2017
Approval date : Feb. 21, 2017

† Department of Computer and Information, Kyungpook National University (E-mail : oh123789@naver.com)
†† Department of Software, Kyungpook National University
※ This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korean government(MSIP) (NRF-2016R1A2B4009793)

한다는 단점이 있고, A*알고리즘은 탐색 범위를 줄일 수 있으나, 가장 짧은 경로를 보장 하지 않는다. 따라서 A*알고리즘에 기반 한 다양한 연구[6]가 진행 되었다.

최근 다양한 방식으로 최단경로 탐색 시간을 단축시키는 방법에 대한 연구들이 진행되었다. 예를 들어, Dijkstra 알고리즘을 이용하여[7,8], 주행시간을 기반으로 경로를 최소화[9]하여 최단거리를 구하는 알고리즘이 있었고, 최단 거리 탐색에 A* 알고리즘과 Dijkstra 알고리즘을 주기적으로 교체하여 서로를 보완하여 만든 A*와 Dijkstra 알고리즘의 하이브리드 검색법[10], 통신과 유전자 알고리즘 기반 동적 차량 경로 탐색 알고리즘[11,12], 선호도 기반 최단경로 탐색을 위한 휴리스틱 융합 알고리즘[13], 양방향에서 검색하는 Bi-directional[14], 여러 개의 소규모 검색 영역을 검색하는 Stochastic time-dependent planning 방식[15], 차량 운행 환경에서 최단 경로 탐색 알고리즘(예: [16,17,18,19,20,21])이 있었다. 이러한 논문들은 최단경로탐색 시간을 감소시키는 것만을 목표로 하였다. 그러나 스마트 자동차에서의 내비게이션에서는 빨리 최단경로 탐색 시간을 단축하는 것도 중요하지만, 교통 상황의 변화(예: 교통사고, 도로 보수공사, 우회로)에 따라서 실시간으로 최단경로가 갱신되는 것이 중요하다. 예를 들어, 교통사고나 도로 보수 공사와 같은 갑작스런 도로 상황의 변화에 따라서 기존의 최단 경로가 수정되어야 하는 경우에, 현재 위치에서 목적지까지 새로운 최단경로를 구하는 것이 필요하다.

본 논문에서는 방향성이 있는 동적인 도로에서 실시간 최단경로 탐색 시스템을 구현하여, 방향성이 있는 도로를 설정하고, 임의의 설정으로 도로의 상황을 변화시킨 뒤, 다양한 조건(즉, 최단 경로를 탐색하는 차량 대수의 변화, 도로 상황 변화, 방향성 있는 도로들의 비율 변화)을 변경하면서, 차량과 서버와의 통신횟수와 도착 시간에 얼마나 영향을 주는지에 대하여 거리기반과 주행시간기반 최단경로 알고리즘을 이용하여 비교 분석하고, 그에 따른 적합한 알고리즘을 제시한다.

2. 관련연구

2.1 Dijkstra 알고리즘

위 알고리즘은 최단 거리 경로를 계산하기 위하여, 우선순위 큐를 이용한 Dijkstra 알고리즘을 정리한 것이다. Dijkstra 알고리즘은 양의 가중치를 갖는 에지(edge)들로 이루어진 방향 그래프에 대해 단일 출발 노드(node)에서 출발하여 다른 모든 노드들까지의 최단 경로를 찾는 알고리즘이다. 그래프는 모든 노드와 노드 간의 거리, 인접한 노드들에 관한 정보를 포함한다. 위 알고리즘에서 source는 출발 노드를 의미하고, INFINITY는 경로가 존재하지 않음을 의미하며, prev[node]는 출발지에서 목적지까지 경로를 저장하기 위하여 이전의 노드를 저장한다. dist[node]는 출발지부터 해당 node까지 최단 거리를 저장한다.

```
function Dijkstra (Graph, Source):
 1:     dist[source] ← 0
 2:     prev[source] ← undefined
 3:
 4:     create vertex set Q
 5:
 6:     for each vertex v in Graph:
 7:         if v ≠ source
 8:             dist[v] ← INFINITY
 9:             prev[v] ← UNDEFINED
10:         add v to Q
11:
12:     while Q is not empty:
13:         u ← vertex in Q with mindist[u]
14:         remove u from Q
15:         for each neighbor v of u:
16:             alt ← dist[u] + length(u, v)
17:             if alt < dist[v]:
18:                 dist[v] ← alt
19                  prev[v] ← u
20:     return dist[], prev[]
```

Fig. 1에서 화살표는 도로의 방향성을 의미하며,

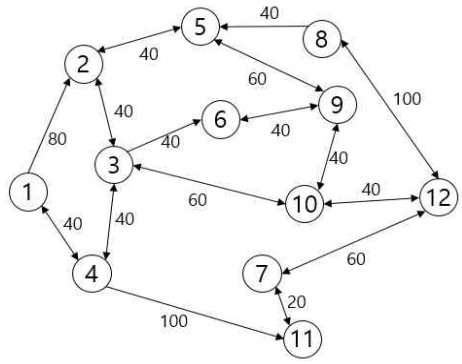


Fig. 1. Example Map.

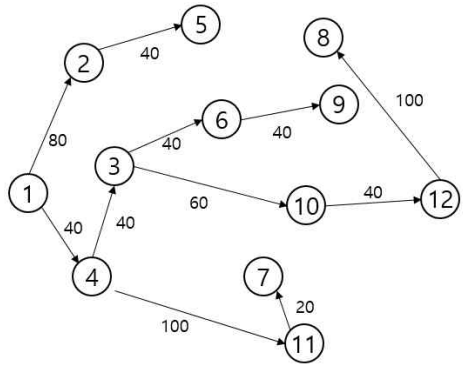


Fig. 2. Shortest paths from node 1 to the others based on Dijkstra algorithm.

화살표 옆의 숫자는 도로의 길이를 의미한다. 원 안에 있는 숫자는 노드 식별자이다. 본 예제에서는 12개의 노드가 존재한다. n 노드와 m 노드사이의 도로는 (n,m)으로 표시한다. 예제 도로는 16개의 에지로 구성되어 있고, (1,2), (1,4), (2,3), (2,5), (3,4), (3,6), (3,10), (4,11), (5,8), (5,9), (6,9), (7,11), (7,12), (8,12), (9,10), (10,12)이 있다. Dijkstra 알고리즘을 적용하여, Fig. 1의 노드 1에서부터의 다른 노드들까지의 최단 거리 경로는 Fig. 2와 같다. 예를 들어 노드 1에서 노드 8로 이동하려면, 최단 거리 경로는 1-4-3-10-12-8이며 이때 최단 거리는 280이다.

2.2 주행시간 기반 최단경로 탐색 알고리즘

주행시간 기반 최단경로 탐색 알고리즘은 방향그래프의 최단거리를 빠르게 탐색하기 위해 3단계에 걸쳐 그래프를 단순화 시켜 최단경로를 계산한다. 노드(Nodes, n)와 호(Arcs, a)의 방향 그래프 $G=(N,A)$일 때, 호의 가중치 (주행시간) = 거리/주행속도 이고, 각 호들에 대해 |n|×|n| 가중치 정방행렬 작성한다. 그리고 출발 노드부터 목적지 노드까지 출발 노드를 레벨 1로 시작하여, 출발노드의 인접노드를 레벨 2, 그 인접노드의 인접노드를 레벨 3, 이러한 방식으로 레벨을 순차적으로 증가시킨다. 그래프 레벨을 생성한 이후에, 단순화를 위하여 다음과 같은 단계를 실행한다.

1단계 P1. 역-레벨 호 삭제
2단계 P2. 노드 최소 가중치 호 선택
3단계 S1. 레벨 단위 최소 주행시간 선택

1단계에서 경로 노드 중 유입차수가 0이고 $(d_i=0)$ 유출차수가 0인 $(d_o=0)$ 노드의 호들 모두 삭제한다.

2단계에서 Table 1과 같은 조건으로 가중치 호를 선택한다.

이때 역시 경로 노드 중 $d_i=0$와 $d_o=0$인 노드의 호들을 삭제한다.

3단계에서 다음과 같은 조건으로 최소 주행시간을 계산한다.

$d_i \geq 2$인 L_i 레벨 경로 노드 y에 대해 L_{i-1} 레벨 노드 x와 L_i 레벨 형제노드(Sibling) z의 L_{i-1} 레벨 노드 x간 비교한다.

(1) (x, z)는 없고 (w, y),(z, y)만 존재하면 (z, y)호 삭제

(2) (w, y),(x, y)(y, z)가 존재하고, (w, y)>(x, z)이면 (y, z)호 삭제

(3) (w, y),(x, z),(y, z)가 존재하면 (y, z)호 삭제

(4) (x, y),(x, z),(y, z),(z, y)가 존재하면 max{(x, y)+(y, z), (x, z)+(z, y)}호 삭제

(5) (x, z)는 없고 (x, y),(y, z)만 존재하면 (y, z)호

Table 1. Level 2 P2

node	row (outflow edge)	column(inflow edge)
Start node (S)	Selection of two edges with minimum weight (if there are multiple edges with minimum weight, select them)	Not applicable
Intermediate node (P)	Selection of one edge with minimum weight (if there are multiple edges with minimum weight, select them)	Selection of one edge with minimum weight (if there are multiple edges with minimum weight, select them)
Destination node (D)	Not applicable	Selection of two edges with minimum weight (if there are multiple edges with minimum weight, select them)

삭제 안함

이때 역시, 경로 노드 중 $d_i = 0$와 $d_o = 0$인 노드의 호들을 삭제한다.

최종 적으로 최소 주행시간 경로 선택은 다음과 같은 조건으로 한다.

IF 모든 노드들이 $d_o \leq 1$와 $d_i \leq 1$

THEN 알고리즘 종료한다.

ELSE IF $d_o \geq 2$ 노드부터 $d_i \geq 2$ 노드까지 경로 최소 주행시간 선택

(만약, 동일 경로 시간은 노드 수가 적은 경로 선택)

$d_i \geq 2$인 노드가 발견되지 않을 때 까지 반복 수행한다.

3. 방향성이 있는 동적인 도로에서 최단경로 탐색

방향성이 있는 동적인 도로에서 실시간 최단경로 탐색 시스템을 설계하고 구현하기 위해서 Fig. 3의 유스케이스 다이어그램으로 사용자 시나리오를 정리하였다. Fig. 3에서 보는 것처럼, 사용자는 목적지 정보를 입력하고, 운행 중에 도로 상황의 변하면 목적지까지 갱신된 최단 경로 경로를 제공받으며, 서버는 목적지 정보, 해당 차량의 위치 정보를 사용하여 도로 상황에 적합한 최단경로를 계산한다.

3.1 방향성 도로

단방향 화살표는 단방향 통행 도로를 의미하고, 양방향 화살표는 양방향 통행이 가능한 도로를 의미

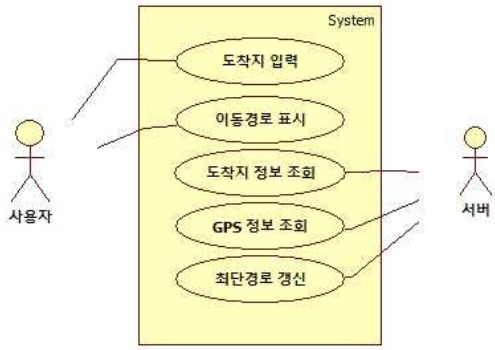


Fig. 3. Use-Case Diagram.

한다. Fig. 1에서 (2,5) 도로를 2→5와 5→2로 나누며 앞으로 에지(edge)라고 부르겠다. 도로의 방향성은 다음과 같은 방법으로 단방향과 양방향으로 구분한다. 단방향 도로는 한쪽 방향에만 속도정보가 존재하고, 양방향은 도로는 양방향 속도정보가 각각 존재한다. 예를 들어, Fig. 3에서 보면 (1,2) 도로는 단방향 도로이고, 1→2 에지는 속도가 40이고, 2→1 에지는 속도가 0이다. (2,5) 도로는 양방향 도로이며, 2→5 에지와 5→2 에지 모두 속도가 60이다.

3.2 사용하는 클래스 소개

각 에지(edge)와 노드(node), 차량(car)은 다음과 같은 정보를 가진다.

* edge(edgeID, startNodeID, endNodeID, dist, time, speed)
* node(nodeID, X, Y, adjoinEdge)
* car(carID, X, Y, hereEdgeID, shortPathDist, shortPathTime, dist, time)

각 에지(edge)는 도로의 한 방향을 의미하며, 시작노드ID, 끝노드ID, 도로 길이, 도로주행속도, 도로주행시간 정보를 가지고 있다. 각 노드(node)는 도로의 끝, 정점을 의미하며, 노드의 X좌표, 노드의 Y좌표와 노드에 접해있는 에지들의 정보를 가지고 있다. 각 차량(car)은 현재위치의 X좌표, 현재위치의 Y좌표, 현재 에지ID, 최단거리경로, 최단시간경로, 최단 거리, 최단 시간 정보를 가지고 있다.

3.3 최단 경로 탐색

최단경로 탐색은 거리기반과 주행시간 기반으로 나누어 실험하였다.

최단경로 중 거리기반 탐색은 Fig. 1과 같이 설정된 에지마다 기본 거리를 기반으로 Dijkstra 알고리즘을 적용하여 최단경로를 구하며, 시간기반 탐색은 $\text{주행시간}(t) = \frac{\text{거리}(dist)}{\text{속력}(speed)}$ 을 이용한다. Fig. 1의 거리정보와 Fig. 4의 기본속도정보를 이용하여 Fig. 5와 같이 에지마다 주행시간을 구하여 저장한 환경에서 Dijkstra 알고리즘을 적용한다.

3.3.1 노드에 인접한 에지

본 논문에서는 빠른 Dijkstra 탐색을 위해 위와 같

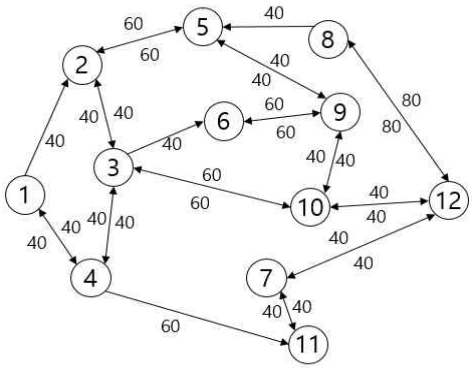


Fig. 4. Example speed of roads.

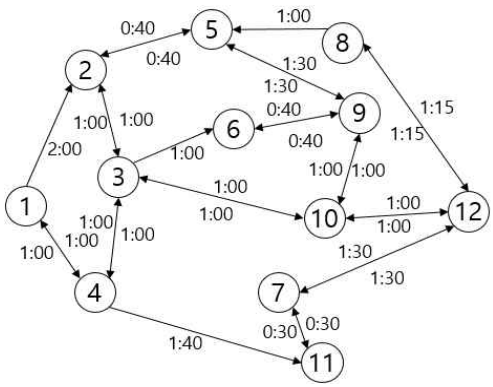


Fig. 5. Example driving time of roads.

```
function FindAdjoinEdge(Graph):
    for each edge e in Graph:
        start ← startNodeID[e]
        end ← endNodeID[e]
        altS ← adjoin[start] + e;
        altE ← adjoin[end] + e;
        adjoin[star] ← altS
        adjoin[end] ← altE
    return adjoin[]
```

은 알고리즘을 이용하여 인접한 노드를 모두 구하였다. start는 에지의 시작노드ID를 저장한다. end는 에지의 끝노드ID를 저장한다. adjoin[node]는 인접한 에지ID를 문자열 혹은 리스트로 저장한다.

3.3.2 거리기반 최단경로 탐색

본 논문에서 사용된 Dijkstra 탐색은 거리 기반과 주행 시간 기반이다. 앞의 우선순위 큐를 이용한 거리 기반 Dijkstra 탐색와의 차이점은 인접한 노드 대신 인접한 에지를 사용하였고, 방향성을 구분하기 위하여 에지 정보에 기본속도를 넣어 갈수 있는지 없는지를 구분하였다. 기본속도가 0이면 사용할 수 없는 도로를 뜻한다. distance(edge)는 edge에서 거리를 정보를 가져온다.

```
function Dijkstra(Graph, Source):
    dist[source] ← 0
    prev[source] ← undefined

    create vertex set Q

    for each vertex v in Graph:
        if v ≠ source
            dist[v] ← INFINITY
            prev[v] ← UNDEFINED
        add v to Q

    while Q is not empty:
        u ← vertex in Q with mindist[u]
        remove u from Q
        for each adjoinedge e of u:
            if speed[e] > 0
                alt ← dist[u] + distance(e)
                if alt < dist[v]:
                    dist[v] ← alt
                    prev[v] ← u
    return dist[], prev[]
```

3.3.3 시간기반 최단경로 탐색

```
function DijkstraTime(Graph, Source):
    time[source] ← 0
    prevT[source] ← undefined

    create vertex set Q

    for each vertex v in Graph:
        if v ≠ source
            time[v] ← INFINITY
            prevT[v] ← UNDEFINED
        add v to Q

    while Q is not empty:
        u ← vertex in Q with mintime[u]
        remove u from Q
        for each adjoiinedge e of u:
            alt ← time[u] + distance(e)
            if alt < time[v]:
                time[v] ← alt
                prevT[v] ← u
    return time[], prevT[]
```

최단 시간 경로를 구하기 위해, 우선순위 큐를 이용한 Dijkstra 알고리즘을 시간 기반으로 정리한 것이다. 그래프는 인접한 노드들에 관한 정보를 포함하는 것과 source는 출발 노드, INFINITY는 무한대로 동일하며, 모든 노드와 노드 간의 거리대신에, 노드와 노드간의 이동하는데 걸리는 주행시간이 포함된다. prevT[node]는 출발지에서 목적지까지 경로를 저장하기 위하여 이전의 노드를 저장한다. time[node]는 출발지부터 해당 node까지의 최단 시간을 저장한다.

3.3.4 최단 경로 갱신

본 논문에서는 동적인 도로에서 실시간으로 최단 경로를 반영하기 위해 상황을 일정한 수치로 변화를 주고, 변화 하였을 때 조건에 만족하는 차량에 최단 경로를 갱신하도록 하였다. 이때 다음과 같은 정보를 사용한다.

* ChangeEdgeList(CedgeID, edgeSpeed, nowSpeed)

ChangeEdgeList는 속도가 변화한 에지의 리스트이며, edgeID는 변화한 에지의 ID이다. edgeSpeed는 에지의 기본속도이다. nowSpeed는 현재 에지의 속도이다.

이 정보들은 다음과 같은 규칙을 따른다.

에지의 현재 속도를 nowSpeed에 저장한다.

에지의 speed(도로기본속도)를 edgeSpeed에 저장한다.

속도가 변화할 때 마다 nowSpeed는 갱신된다.

edgeSpeed와 nowSpeed가 다르면, ChangeEdgeList에 추가한다.

ChangeEdgeList가 발생하면, 에지의 dist와 nowSpeed를 이용하여 에지의 time(도로주행시간)을 갱신한다.

edgeSpeed와 nowSpeed가 일치하면, ChangeEdgeList에서 삭제된다.

Fig. 6은 최단 시간 경로를 구할 때, 정체가 발생하여 ChangeEdgeList가 발생한 것을 보여주는 예시이다. Fig. 4에서 3→10에지는 속도가 30이 되었고, 9→10에지의 속도는 20으로 되어, Fig. 6의 도로 주행

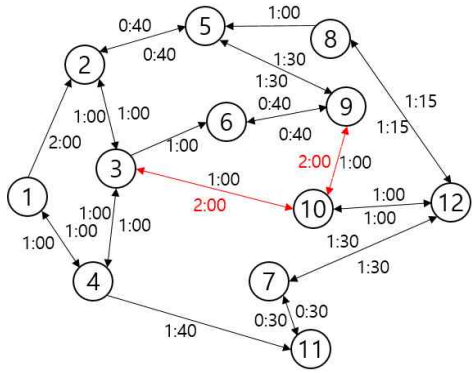


Fig. 6. Example 3→10, 9→10 road traffic congestion.

속도가 느려진 것을 보여준다. 그 때 속도와 각 애지의 거리를 참고로 주행 시간이 3→10에지는 60/30을 하여 2시간이 되고, 9→10에지 40/20을 하여 2시간이 되어, 각 도로가 갱신되었다.

차량의 최단 경로는 ChangeEdgeList에 갱신이 있을 때 마다 포함되어 있는 에지가 최단 경로에 포함되어 있지 여부를 확인하며, 만약 포함 되어 있을 경우 차량의 최단경로는 현재 위치에서부터 목적지까지 새로이 갱신한다. 정보 갱신 순서는 Fig. 7과 같다.

4. 실 험

4.1 실험환경

Fig. 10은 실험에서 사용한 지도를 보여준다. 지도

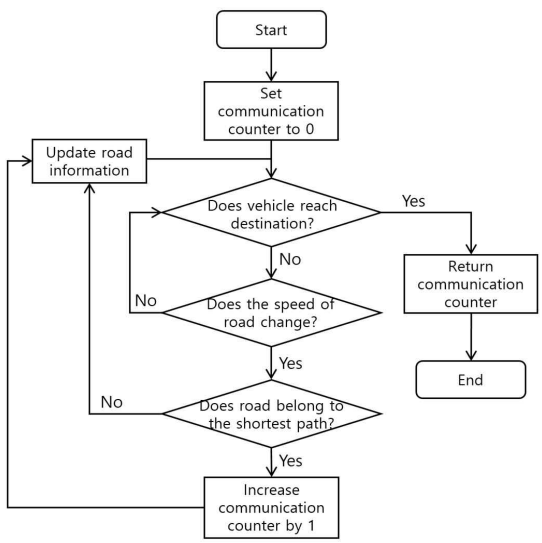
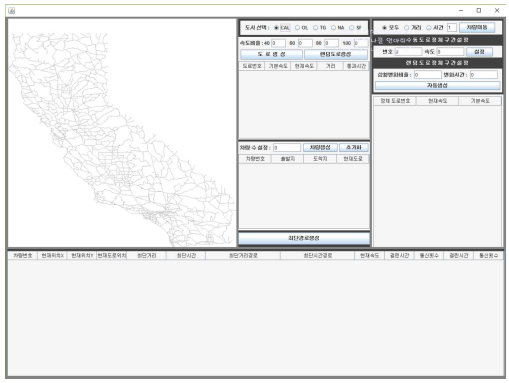


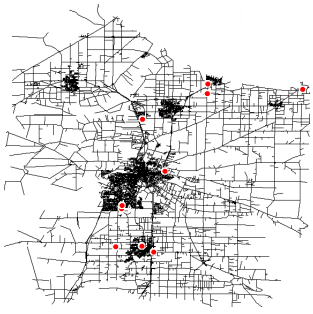
Fig. 7. Flowchart for updating information.

(a) California (b) Oldenburg (c) San Joaquin (d) San Francisco

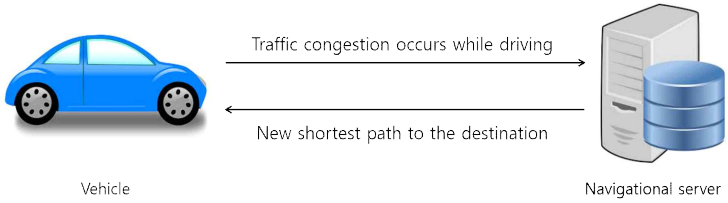
Fig. 8. real-life road data.



(a) Snapshot for running program



(b) Moving vehicles



(c) New shortest path

Fig. 9. Moving vehicles and generation of new shortest paths for them

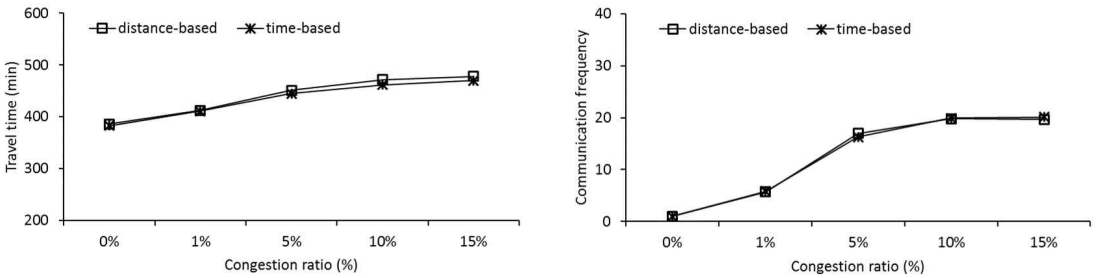
데이터는 [22]에서 받았고, 비교 실험을 위하여 자바 언어와 MySQL DBMS을 사용하여 거리 기반 최단 경로 탐색 방법과 시간 기반 최단경로 탐색 방법 구현하였다. 실험 구동 환경은 Intel i7 CPU와 16 GB 메모리를 가진 컴퓨터에서 이루어졌다. 구현한 프로그램은 전체 도로의 방향성 도로들을 무작위로 생성하여, 차량의 대수를 입력하면 무작위로 출발지와 목적지가 정하고, 그리고 교통 상황 변화율을 반영하기 위하여 정체구간을 비율에 맞게 생성할 수 있는 프로그램이다. Fig. 9(a)에서 보는 것처럼, 구현된 시스템에서 개별 차량이 이동하는 모습을 관찰할 수 있으며, 차량들의 정보는 표로 표시한다. Fig. 9(b)에서 보는 것처럼, 차량들이 이동을 하면, 지도에서 빨간 점으로 표시하였고, Fig. 9(c)에서 보는 것처럼, 운행 중에 교통 정체가 발생하면, 서버에 새로운 경로를 요청한다.

4.2 실험결과

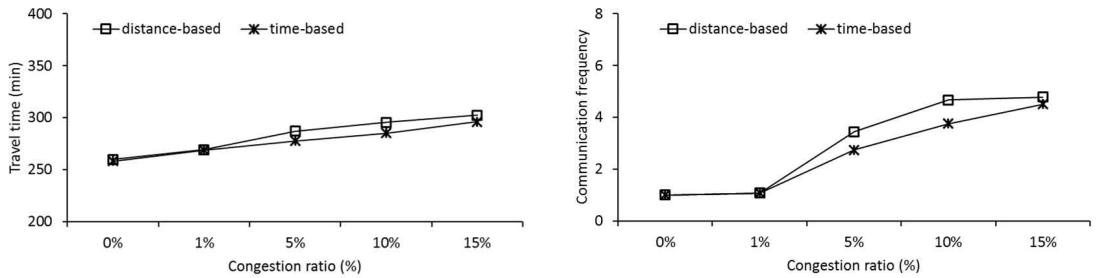
본 논문에서는 교통 상황의 변화에 따른 도착 시

간과 통신 횟수의 변화를 조사하는데 중점을 두고 실험을 진행하였다. 교통 상황의 변화율이란 도로 상황의 변화 정도 (즉, 도로 정체)를 나타내며, 전체 도로의 1%, 5%, 10%, 15%의 경우에 대하여 실험하였다. 아래의 그림들은 도로 속도 분포는 무작위로 지정하였고, 주어진 혼잡 비율에 따라 도로의 운행 가능한 속도를 변하게 만들어 나온 실험 결과의 평균을 측정하였다.

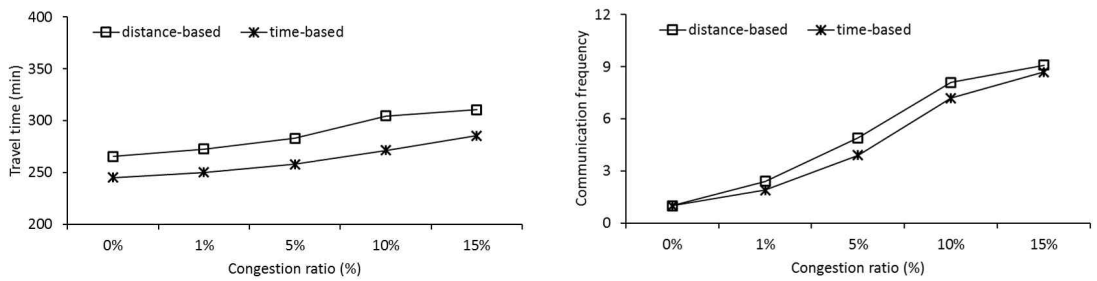
Fig. 10은 다양한 지도 데이터에 대한 거리 기반 최단경로 탐색 방법과 시간 기반 최단경로 탐색 방법의 평균 도착시간과 통신 횟수의 변화를 보여준다. 실험 결과는 혼잡 비율을 0%에서 15%까지 변경하면



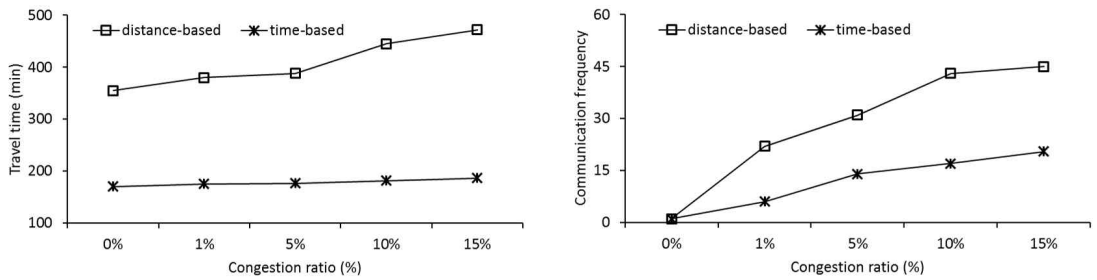
(a) California



(b) Oldenburg



(c) San Joaquin



(d) San Francisco

Fig. 10. Performance evaluation for real-life road data.

서 두 방법의 성능 차이를 조사하였다. Fig. 10(a)는 캘리포니아 지도에서 상황변화를 주었을 때의 평균 도착시간과 통신 횟수의 변화를 보여준다. 상황변화율이 높을수록 도착시간과 통신 횟수는 함께 증가하였다. 재미있는 것은 혼잡비율이 10%에서 15%로 증가할 때, 통신 횟수는 크게 증가하지 않았다. Fig. 10(b)는 올댄버그 지도에서 상황변화를 주었을 때의 평균 도착시간과 통신 횟수의 변화를 보여준다. 목적지까지 운행 시간은 거리 기반 최단 경로 탐색 방법과 시간 기반 최단 경로 탐색 방법에서 주목할 만한 차이가 나타나지 않았다. 거리 기반 최단 경로 탐색 방법이 시간 기반 최단 경로 탐색보다 많은 통신 횟수를 보여주었다. Fig. 10(c)는 샌 호아 킨 지도에서 상황변화를 주었을 때의 평균 도착시간과 통신 횟수의 변화를 보여준다. 마지막으로, Fig. 10(d)는 샌프란시스코 지도에서 상황변화를 주었을 때 도착시간과 통신 횟수이다. 인접한 에지가 많아서 전체적으로 기울기가 작다. 위 그림들의 그래프를 보면 대부분 거리기반 최단경로 탐색 방법보다 시간기반 최단경로 탐색방법이 목적지까지 빠르게 도착했으며, 통신 횟수가 적다는 것을 보여준다. 혼잡 비율이 높은 경우에, 인접한 도로가 많을수록 통신 횟수와 도착시간의 변화가 적었다. 당연히, 정체되는 도로가 많을수록 목적지까지 도착시간은 늦어지며, 통신 횟수가 증가하는 것을 확인할 수 있다.

5. 결 론

본 논문에서는 방향성이 있는 동적인 도로에서 실시간 최단 경로 탐색 시스템을 설계하고 구현하였다. 실험에서는 실제 도로 데이터에 방향성을 부여하고, 도로의 정체 상황을 만들고, 운행 환경의 변화에 따라 실시간으로 최단 경로를 탐색하는 시스템을 설계하고 구현하였다. 다양한 도로 상황 변화를 고려하기 위하여, 도로에 혼잡 비율을 0%에서 15%까지 변경하면서, 거리기반 최단경로 탐색 방법과 시간기반 최단경로 탐색 방법으로 구현하여 목적지까지 운행시간과 통신 빈도를 비교하였다. 도로의 상황변화율에 따라 도착 시간과 통신 횟수에 얼마나 영향을 주는지 확인하였다. 실험 결과에서 도착 시간과 통신 횟수가 반드시 비례하지 않는다는 것도 보여 주었다. 실험 결과를 요약하면, 거리기반 탐색 방법보다 시간기반 탐색 방법이 목적지까지 빠르게 도착하는 경우가 많았고, 통신 횟수가 적다는 것을 보여주었다. 혼잡한 도로 환경에서 우회할 수 있는 도로가 많을수록 통신 횟수와 목적지까지 도달하는 시간의 변화가 작았다. 바꾸어 말하면, 혼잡한 도로 환경에서 차량들이 목적지까지 도달하는 시간이 길어졌고, 통신 횟수가 증가하였다.

REFERENCES

[1] B.H. Wellenhof, H. Lichtenegger, and J. Colins, *Global Positioning System: Theory and Practice*, Springer-Verlag, New York, 2001.

[2] E.W. Dijkstra, "A Note on Two Problems in Connextion with Graphs," *Numerische Mathematik*, Vol. 1, No. 1, pp. 269-271, 1959.

[3] P.E. Hart, N.J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on System Science and Cybernetics*, Vol. 4, No. 2, pp. 100-107, 1968.

[4] M. Abboud, L. Mariya, A. Jaoude, and Z. Kerbage, "Real Time GPS Navigation System," *Proceeding of the 3rd FEA Student Conference*, pp 1-6, 2004.

[5] T.H. Cormen, C.E. Leserson, R.L. Rivest, and C. Stein, Introduction to Algorithms 3rd Edition, MIT Press and McGraw-Hill, Cambridge, Massachusetts, 2009.

[6] L.H.O. Rios and L. Chaimowicz, "A Survey and Classification of A* Based Best-First Heuristic Search Algorithms," *Proceeding of the 20th Brazilian Symposium on Artificial Intelligence*, pp. 253-262, 2010.

[7] Y.T. Lim and H.M. Kim, "A Shortest Path Algorithm for Real Road Network based on Path Overlap," *Journal of the Eastern Asia Society for Transportation Studies*, Vol. 6, pp. 1426-1438, 2005.

[8] F.B. Zhan, "Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures," *Journal of Geographic Information and Decision Analysis*,

Vol. 1, No. 1, pp. 69-82, 1997.

[9] S.U. Lee, "A Real-time Point-to-Point Shortest Path Search Algorithm Based on Traveling Time," *Journal of The Institute of Internet, Broadcasting and Communication*, Vol. 12, No. 4, pp. 131-140, 2012.

[10] Y. Lee and S. Kim, "A Hybrid Search Method of A* and Dijkstra Algorithms to Find Minimal Path Lengths for Navigation Route Planning," *Journal of the Institute of Electronics and Information Engineers*, Vol. 51, No. 10, pp. 109-117, 2014.

[11] C.W. Ahn, R.S. Ramakrishna, and C.G. Kang, "A New Genetic Algorithm for Shortest Path Routing Problem," *Journal of the Korean Institute of Communications and Information Sciences*, Vol. 27, No. 12C, pp. 1215-1227, 2002.

[12] B. Oh, J. Bae, J. Yang, and J. Nang, "Genetic Algorithm-based Dynamic Vehicle Route Search using Car-to-Car Communication," *Advances in Electrical and Computer Engineering*, Vol. 10, No. 4, pp. 81-86, 2010.

[13] S.H. Ok, J.H Ahn, S.H Kang, and B.G Moon, "A Combined Heuristic Algorithm for Rreference-based Shortest Path Search," *Journal of the Institute of Electronics Engineers of Korea TC*, Vol. 47, No. 8, pp. 74-84, 2010.

[14] J.M. Lee, J.H. Kim, and H.S. Jeon, "Performance Evaluation of Different Route Planning Algorithms in the Vehicle Navigation System," *Journal of Korean Association of Information Education*, Vol. 2, No. 2, pp. 252-259, 1998.

[15] I.C.M. Flinsenberg, *Route Planning Algorithms for Car Navigation*, Ph.D Thesis of Eindhoven University, 2004.

[16] S.H. Ok, J.H. Ahn, S. Kang, and B. Moon, "A Combined Heuristic Algorithm for Preference-based Shortest Path Search," *Journal of the Institute of Electronics Engineers of Korea TC*, Vol. 47, No. 8, pp. 74-84, 2010.

[17] B.W. Lee, W.K. Choi, and H.T. Jeon, "Intelligent Navigation System Using Fuzzy Logic," *Journal of the Institute of Electronics Engineers of Korea CI*, Vol. 43, No. 4, pp. 67-72, 2006.

[18] L. Qi and M. Schneider, "Realtime Response of Shortest Path Computation," *Proceeding of the 7th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pp. 41-46, 2014.

[19] L. Hu, J. Yang, and J. Huang, "The Real-time Shortest Path Algorithm with a Consideration of Traffic-light," *Journal of Intelligent and Fuzzy Systems*, Vol. 31, No. 4, pp. 2403-2410, 2016.

[20] D. Gupta, U. Dutta, and P. Gupta, "An Algorithm and Evaluations of Real Time Shortest Path According to Current Rraffic on Road," *Journal of Scientific Engineering and Applied Science*, Vol. 1, No. 7, pp. 279-284, 2015.

[21] S.M. Gang, D.H. Ryu, T.S. Kim, H.C. Park, J.M. Kim, and Y.J. Choung, "Development of Core Module and Web System for a Visualization Platform for the 3D GIS Service of Disaster Information using Unity," *Journal of Korea Multimedia Society*, Vol. 20, No. 3, pp. 520-532, 2017.

[22] Real Datasets for Spatial Satabases, http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm, (accessed Feb., 21, 2017).

권 오 성

2015년 경북대학교 컴퓨터정보학부 학사
2017년 경북대학교 컴퓨터정보학과 공학석사
관심분야 : 데이터베이스, 위치기반서비스, 지리정보시스템

조 형 주

1997년 서울대학교 컴퓨터공학과 학사
1999년 서울대학교 컴퓨터공학과 공학석사
2005년 한국과학기술원 전자전산학과 공학박사

현재 경북대학교 소프트웨어학과 조교수
관심분야 : 데이터베이스, 위치기반서비스, 지리정보시스템